VMEbus en FPGA.

F. E. Achilli,
Facultad de Ingeniería
UNLP
Centro de Técnicas
Analógico-Digitales
(CeTAD)
La Plata, Argentina
eachilli@gmail.com

S. A. Gil.
Facultad de Ingeniería
UNLP
Centro de Técnicas
Analógico-Digitales
(CeTAD)
La Plata, Argentina
agil@gmail.com

J. A. Rapallini,
Facultad de Ingeniería
UNLP
Centro de Técnicas
Analógico-Digitales
(CeTAD)
La Plata, Argentina
josrap@gmail.com

A. A Quijano
Facultad de Ingeniería
UNLP
Centro de Técnicas
Analógico-Digitales
(CeTAD)
La Plata, Argentina
a.quijano@gmail.com

***Resumen* -- Se presenta el diseño del modulo de entada- salida del sistema de comunicación de datos VMEbus, cumpliendo con las especificaciones definidas en el estándar IEEE 1014-87 Revisión B..**

En el desarrollo se utilizó estándares de comunicación, lenguajes de descripción de hardware, herramientas para el diseño de hardware-software, instrumental para la verificación del diseño y dispositivos lógicos programables para la implementación.

El trabajo presenta el marco teórico, realiza la descripción general de la interfase y en particular el detalle del desarrollo de los bloques implementados, dando los conceptos necesarios para la comprensión del diseño en VHDL y su implementación en FPGA.

Palabras claves: diseño digital, FPGA, VMEbus

I. INTRODUCCIÓN

Entre los sistemas de comunicación de datos instrumentales de alta perfomance se encuentra el VMEbus (IEEE-1014-87), su arquitectura es independiente del microprocesador que lo utilice y posee configuración dinámica de bus de datos (8 a 32 bit) y de bus de direcciones (16, 24 o 32 bit); también presenta características de arquitectura maestro esclavo, capacidad multiprocesamiento (1 a 21), alta transferencia de datos (típico 40 Mb/seg.) y otras propiedades que aun lo mantienen en vigencia como interfaz entre equipos de instrumental científico.

A. Características del VMEbus

- Comunicación de dispositivos conectados al bus sin perturbación de las actividades internas de otros dispositivos conectados al mismo.
- Requerimientos eléctricos y mecánicos para diseñar dispositivos que sean capaces de comunicarse con otros dispositivos que se encuentren conectado al bus.
- Protocolos de interacción entre el bus y los dispositivos conectados a él.
- Proveer definiciones y terminología que describan el protocolo.
- Permitir un amplio rango de diseño, donde se pueda optar por optimizar costo ó performance, ó ambos, sin afectar la compatibilidad del sistema.
- Proveer un sistema donde la performance es principalmente limitada por el dispositivo antes que limitada por el sistema de interfase.

B. Especificaciones del VMEbus

El diseño se realizó en base a lo descrito por las especificaciones VMEbus (IEEE-1014-87 revisión B.), por lo tanto, todas las especificaciones, características y definiciones mencionadas en este trabajo corresponden a dicha revisión.[1] [2] [3] [4] [5]

El VMEbus utiliza una arquitectura Maestro-Esclavo, en la que los módulos funcionales llamados Maestros transfieren datos desde y hacia los módulos llamados Esclavos. El estándar permite que varios Maestros residan sobre el mismo bus, es por eso que a este se lo denomina multiprocesador. Esta configuración implica que antes de realizar una transferencia de datos, el Maestro previamente deberá adquirir el control del bus mediante un proceso llamado arbitración.

Toda la actividad efectuada sobre el bus se realizará sobre cuatro sub-buses: el bus de transferencia de datos, el bus de arbitración para la transferencia de datos, el bus de prioridad de interrupciones y el bus de utilidades. Cada uno de estos cumple una función específica dentro del protocolo VME.

El VMEbus (Tabla 1) es un bus paralelo asincrónico (es decir, que no utiliza reloj para sincronizar las transferencias de datos). La velocidad de transferencia está dada por la velocidad del módulo más lento que participa en el ciclo, siendo la velocidad de transferencia máxima de 40 MBytes/s. Además dispone de un sistema de interrupciones con siete niveles de prioridad

TABLE I. CARACTERÍSTICAS GENERALES DEL VMEBUS

Item	Especificación	Nota
Arquitectura	*Maestro / Esclavo*	
Mecanismo de Transferencia	*Asincrónico, no multiplexado*	*Sin reloj de sincronización central*
Rango de direccionamiento	*16-bit (short I/O)* *24-bit (estándar)* *32-bit (extended)*	*Selección dinámica*
Ancho de dato	*8, 16, 24 ó 32 bit*	*Selección dinámica*
Transferencia de datos no alineados	*si*	*Compatible con la mayoría de los microprocesadores*
Detección de errors	*si*	*Uso de BERR* (opcional)*
Rango de transferencia de datos	*0-40 Mbyte/sec*	
Interrupciones	*7 niveles*	*Sistema de interrupción con prioridad*
Capacidad de multiprocesamiento	*1-21 procesadores*	*Arbitración de bus flexible*
Sistema de capacidad de diagnostico	*si*	*Uso de SYSFAIL* (opcional)*
Normas mecánicas	*Simple* *Doble*	*160 x 100 mm eurocard* *160 x 233 mm eurocard* *conectores DIN 603-2*
Normas internacionales	*si*	*IEC 821, IEEE 1101, IEEE 1014*

C. Elementos del sistema de interfase

La estructura del sistema puede ser descripta a través de la estructura mecánica y funcional del mismo. Las especificaciones mecánicas describen las dimensiones físicas del backplane, conectores, chasis, etc. Las especificaciones funcionales describen la manera en la que el bus opera, los módulos funcionales que están involucrados en cada transacción y las reglas que gobiernan su comportamiento.

D. Estructura Básica

La estructura funcional está dividida en cuatro categorías. Cada una de estas consiste en un bus y módulos funcionales asociados que trabajan en conjunto para realizar tareas específicas. En la Figura 1 se muestra la estructura básica del sistema VME, el bloque de la izquierda corresponde al Módulo de Control del Sistema, este módulo es el encargado administrar los recursos del sistema; el bloque del centro corresponde al Módulo de Procesamiento, este inicializa los ciclos de transferencia de datos para transferir información entre él mismo y los distintos dispositivos de entrada-salida y por último el bloque de la derecha es el Módulo de Entrada-Salida, que será el que se desarrolla en este trabajo.

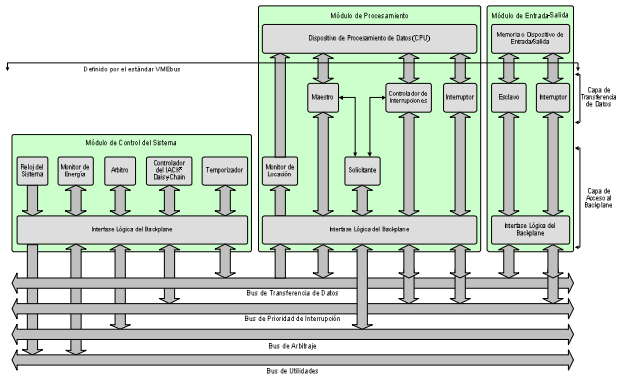

Figura 1.– Diagrama en bloques funcionales del VMEbus

II. DISEÑO DE LA INTERFASE VMEBUS-ESCLAVO

Todo el diseño se realizo con modulos en VHDL [6] [7] [8], del los cuales se presentan los detalles de funcionamiento y desarrollo de sus bloques funcionales.

El estándar permite varios tipos de ciclos de bus: lectura/escritura, transferencia en bloque, lectura-modificación-escritura y solo-direccionamiento. Los requerimientos para este diseño incluyen a todos estos, a excepción del ciclo de transferencia en bloques. El Interruptor se eligió del tipo D08(O) ROAK por ser soportado por todos los Controladores de Interrupciones.

A pesar de que el VMEbus es un bus asíncrono, la programación de los bloques se realizará de forma síncrona, es decir, una señal de reloj comandará a dichos bloques. Esto se basa en las ventajas que presenta frente al diseño asíncrono, algunas de las cuales son: muestreo de las señales en instantes de tiempo conocidos y bien definidos, mayor inmunidad a los *glitches* y simplificación del problema de la variación de retardos en los diferentes caminos de la lógica.

La mayoría de las señales definidas en el estándar VMEbus se activan en estado bajo ó con un flanco descendente, debido a esto y como regla mnemotécnica les antepondremos la letra **n** al nombre de las señales en cuestión.

A. Capa de Transferencia de Datos

Esta capa es la encargada de realizar toda la lógica de la interfase, maneja tanto las transferencias de datos entre periféricos y el bus, así como las interrupciones de dichos periféricos

El diseño de esta capa se dividió en dos bloques tal como se muestra en la Figura 2, teniendo cada uno de ellos funciones bien definidas:

1) Esclavo

La función de este bloque consiste en participar de

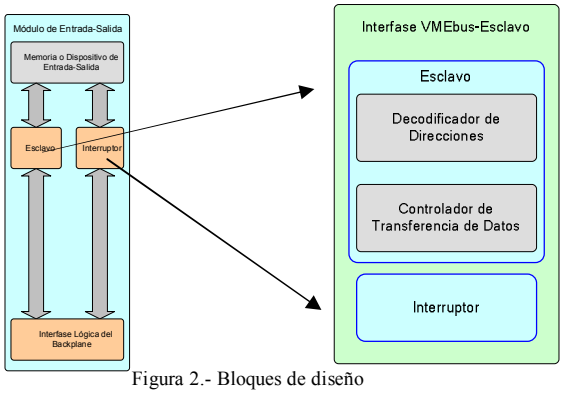

Figura 2.- Bloques de diseño

los ciclos de transferencia de datos que genera el Maestro en posesión del bus. Para esto debe realizar dos funciones claramente diferenciables:

Decodificar las direcciones, con lo cual puede determinar si el periférico debe o no participar del ciclo de transferencia de datos en curso.

Realizar la transferencia de datos, en caso de que el Maestro disponga el intercambio de información con el periférico, dicho intercambio debe producirse de acuerdo a lo especificado en el estándar VMEbus. Estas dos tareas se dividen en dos bloques llamados Decodificador de Direcciones y Controlador de transferencia de Datos

a) Decodificador de Direcciones:

La función de este bloque consiste en monitorear las líneas de control y de direcciones para reconocer cuando el Maestro que esté en posesión del bus desee realizar una transferencia de datos con el Esclavo. El bloque compara las direcciones provenientes del Maestro que se propagan por el bus con las que tiene asignadas el periférico, en caso de que estas direcciones coincidan se le informará de tal situación al bloque de Control de Transferencia de Datos mediante la activación de la señal Slave Selected

Este bloque está constituido por una máquina de estados de tres estados, tal como se puede observar en la Figura 3., cuando esto ocurre la salida pasa a estado bajo y se regresa al estado inicial a la espera de un nuevo ciclo, además se ve que al pasar del segundo o del tercer estado al estado inicial cuando ocurre un flanco ascendente de la señal nAS, con esto se logra que el dispositivo soporte el Ciclo de Sólo-Direccionamiento (*Address Only)* especificado en el estándar VMEbus.

b) Controlador de Transferencia de Datos

La función de este bloque es realizar la transferenci a de datos entre el bus de datos VME y el lado accesible al usuario. Presentan dos entradas y dos salidas de datos de 32 bits cada una, una entrada y una salida servirán para la transferencia de datos con el backplane. Cuando el Decodificador de Direcciones lo habilite, y dependiendo de las señales de control, se transferirán los datos entre dichos puertos según lo definido en el estándar

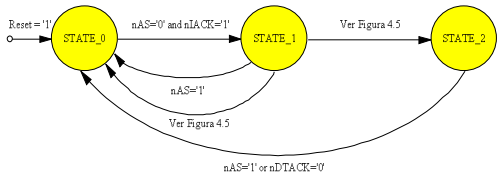

Figura 3 – Diagrama de Estados del decodificador de direcciones

Este bloque está construido por una máquina de estados de cuatro estados la cual puede observarse en la Figura 4

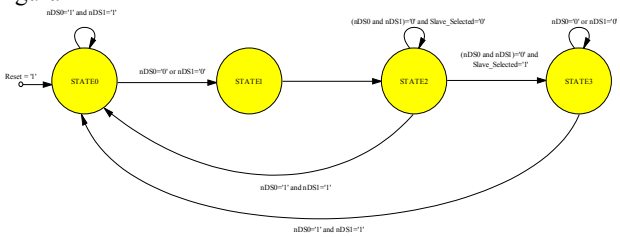

Figura 4 – Diagrama de Estados del Controlador de Transferencia de Datos

c) Puertos Bidireccionales:

En el diseño del Controlador de Transferencia de Datos se necesitan puertos bidireccionales. Si bien en el lenguaje VHDL están contemplados los pines bidireccionales, este bloque tiene entradas y salidas de datos independientes tanto para el lado accesible desde el bus VME como así también para el lado accesible por el usuario.

En la Figura 5 se indica la conexión entre bloques, sus entradas y sus salidas.

2. Interruptor

Una Interrupción es una señal de hardware recibida por el procesador, esta le indica que debe "interrumpir" el curso de ejecución que esta realizando en ese momento y pasar a ejecutar un código específico para tratar una situación particular.

En el VMEbus los Módulos Funcionales son los Interruptores que solo pueden usar una de las siete líneas de pedido de interrupción que tiene el VMEbus y para ello, monitorea las tres líneas más bajas del bus de direcciones (A01-A03) y la línea IACKIN* (para determinar cuando su interrupción será atendida). Cuando esto ocurre, el Interruptor coloca el vector de interrupciones STATUS/ID sobre el bus de datos y le informa al Controlador que es un vector de interrupciones válido activando el DTACK* en bajo.

VMEbus acepta dos clases de Interruptores: los denominados ROAK (Realise-On-Acknowledge) y los denominados RORA (Realise-On-Register-Access). El Interruptor ROAK desactiva su línea de solicitud de interrupción en respuesta al ciclo de reconocimiento de la misma. El mecanismo ROAK funciona con todos los

Controladores. El Interruptor RORA abandona su solicitud cuando el Controlador accede a un registro interno durante la rutina de servicio de interrupción.

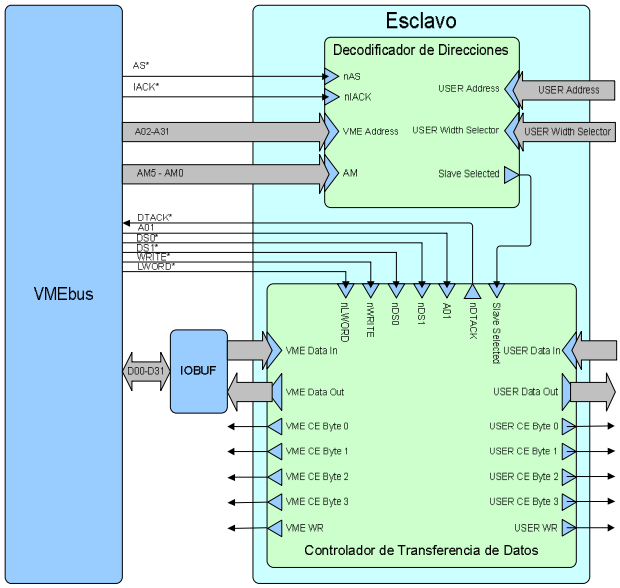

Figura 5: Conexión entre bloques, sus entradas y sus salidas.

Existen distintos tipos de Interruptores en cuanto a opciones de STATUS/ID se refiere. Los Interruptores D08(O) responden a ciclos de pedido de Interrupción de 8, 16 y 32 bits. Estos Módulos ponen un vector de interrupciones (STATUS/ID) de 8 bits sobre las líneas del bus de datos D00-D07. Los Interruptores D16 responden a ciclos de pedido de Interrupción de 16 y 32 bits.

En la Figura 6 se pueden apreciar todas las líneas conectadas al bus que permiten la comunicación con el Controlador de Interrupciones y las líneas provenientes del periférico en turno. Estas son las que nos indican que nivel de interrupciones va a manejar dicho periférico (S1-S3), la señal IRQX que es la señal de interrupción generada por el mismo y el vector de interrupciones (STATUS/ID) que el periférico colocara en el bus.

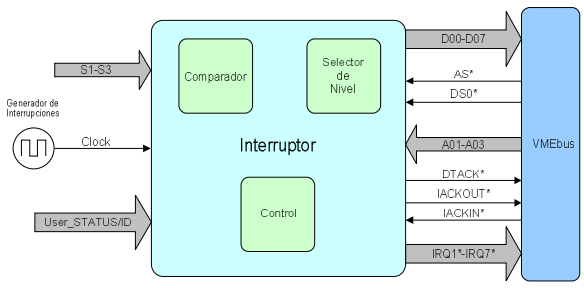

Figura 5: Conexión líneas de entrada –salida del Interruptor.

El módulo Interruptor se diseñó mediante la programación de tres bloques más pequeños que permiten disminuir la complejidad de un bloque único, estos son: el Comparador de Nivel, el Selector de Nivel y por último el bloque de Control.

3) Comparador de Nivel

La función de este bloque es la de definir si el nivel de interrupción que está atendiendo el sistema es el mismo que tiene asignado el periférico. Para esto, monitorea las tres líneas más bajas del bus de direcciones A01-A03 y las compara con las líneas de selección S1-S3 que indican el nivel de interrupción que va a manejar dicho periférico. Cuando el nivel representado por las líneas S1-S3 sea el mismo que el asignado en el bus, entonces se activará la señal Enable_INT.

4) Selector de Nivel

Como se ha mencionado anteriormente, el diseño del Interruptor debe ser capaz de manejar interrupciones de cualquier nivel. La función de este bloque es la de redireccionar la señal de interrupción IRQX proveniente del periférico y situarla sobre el bus en el nivel que corresponda. Para ello monitoreará las líneas selectoras S1-S3 y dependiendo del valor de estas tomará la decisión. Por otro lado el bloque contará con la señal Release_INT, esta es una señal proveniente del bloque de Control que se activa cuando se necesita liberar el pedido de interrupción. En consecuencia, cuando esta señal se active se liberará la señal de interrupción que se este posicionando sobre el bus VME.

5) Control

La función de este bloque es la de definir si la interrupción que se está atendiendo pertenece o no al periférico en turno. El bloque monitorea las líneas nIACKIN, nAS, nDS0 y Enable_INT y una vez que las señales de control provenientes del bus le informan que el Controlador de Interrupciones ha detectado un pedido de interrupción, realizará lo siguiente:

- Verificará si el periférico que esta controlando ha generado una interrupción.
- Si esto ocurre, monitoreará la señal Enable_INT. Si está activa es porque el nivel del periférico y el nivel del pedido de interrupción que detectó el Controlador de Interrupciones es el mismo.
- Dadas estas condiciones se verifica que el pedido de interrupción fue efectuado por el periférico, entonces, se procederá a colocar sobre el bus de datos (D00-D07) el vector de interrupciones STATUS/ID, se activará la señal Release_INT pidiéndole al bloque Selector de Nivel que libere la línea con la cual se ha realizado el pedido de interrupción. Por último, terminará el ciclo activando la señal nDTACK.

- En caso de que los puntos uno y dos no se verifiquen, el pedido de interrupción fue realizado por otro dispositivo. Por lo que se activa la salida nIACKOUT, continuando la cadena Daisy-Chain a lo largo del bus.

6) Interconexión del bloque Interruptor

En la Figura 7 se puede apreciar como se relacionan internamente los distintos bloques del interruptor.

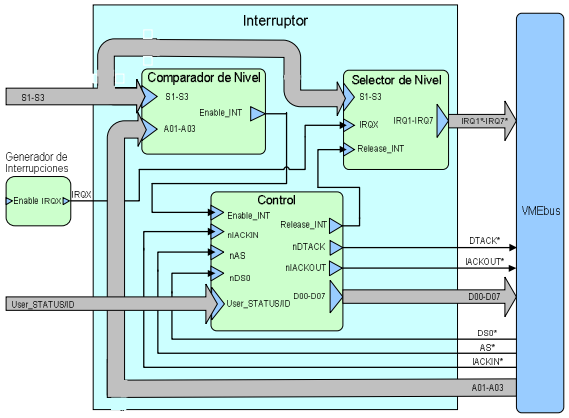

Figura 7: Bloques internos del Interruptor.

III. IMPLEMENTACIÓN

La implementación se realizó sobre un kit de desarrollo con FPGA - Spartan 3 de Xilinx [9], utilizado las herramientas correspondientes [10] [11] [12] para programar y verificar el sistema. Los códigos en VHDL están disponibles en [13].

El estándar VME, plantea requerimientos de tiempo entre señales de pocas decenas de nanosegundos (10, 20, 30 ns), por lo que es necesario contar con un reloj lo suficientemente rápido para poder cumplir estos requerimientos. El reloj que dispone el kit de desarrollo es de 50 MHz, por lo que su período es de 20 ns. Esta frecuencia no impide que el diseño funcione, pero no se logra que el mismo funcione de la manera más eficientemente posible. Por ejemplo no es posible contabilizar un tiempo de 30 ns con un reloj de esta frecuencia.

Cuando se sintetiza un diseño con la herramienta ISE, está brinda un reporte de los aspectos más importantes del hardware generado y realiza un análisis temporal del mismo, el reporte aportado por el ISE dio como resultado:

```
=========================================================================
--Timing constraint: Default period analysis for Clock 'Clock'
--  Clock period: 7.927ns (frequency: 126.151MHz)
--  Total number of paths / destination ports: 4391 / 455
-------------------------------------------------------------------------
```

El resultado muestra que la máxima frecuencia de trabajo para este diseño es de 126 MHz, lo que permite utilizar una frecuencia más veloz de la que brinda el cristal del kit y así lograr un mejor comportamiento del diseño en función de los requerimientos del estándar. Por este motivo se eligió que la frecuencia de trabajo sea de 100 MHz, para lo cual se aprovechó el DCM que dispone la Spartan 3 de Xilinx.

El diseño de la interfase se planteo para ser implementado como se muestra en la Figura 8. Como puede observarse se necesitaran 85 líneas para comunicarse con el VMEbus y 77 líneas para comunicarse con el Esclavo (Periférico).

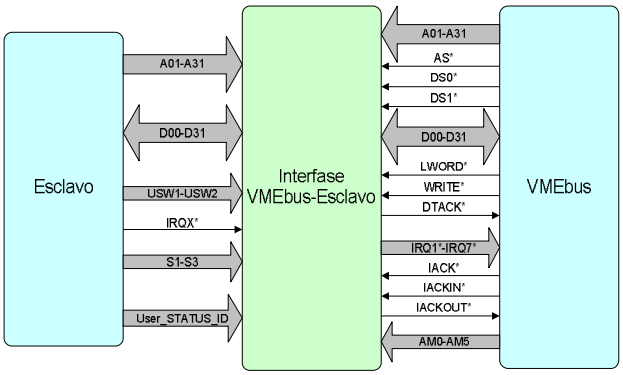

Figura 8: Esquema final diseñado de la Interfaz.

Este diseño al ser implementado en un kit, presentó la imposibilidad del uso de todas las líneas de entrada salida planteadas, ya que *sólo deja 100 pines libres para que el usuario disponga*, por este motivo se debieron analizar soluciones alternativas. Se decidió disminuir la cantidad de líneas a utilizar. Esto solo era posible del lado del esclavo ya que del lado del bus la utilización de las líneas era imprescindible para comunicarse con otro dispositivo VME. Por lo tanto se analizó la posibilidad de implementar la comunicación serie ya que el kit posee el hardware necesario.

Si bien esta solución implica una pérdida en la velocidad de transmisión del VME, se consideró aceptable en el momento del desarrollo, ya que la prioridad era lograr la comunicación entre el bus y el esclavo, más allá de la velocidad de la misma ver Figura 9.

Para poder establecer la comunicación serie entre la interfaz VMEbus-Esclavo y una PC se diseñó la lógica de adaptación que se muestra en la Figura 10. Esta consta de dos partes: la UART (*Universal Asynchronous Receiver-Transmitter*, Receptor-Transmisor Asíncronico Universal) que tiene la función de convertir los datos recibidos en forma

paralela, a forma serial, con el fin de comunicarse con otro sistema externo, realizando también el proceso inverso. La segunda parte consiste en una interfase entre el diseño y la UART.

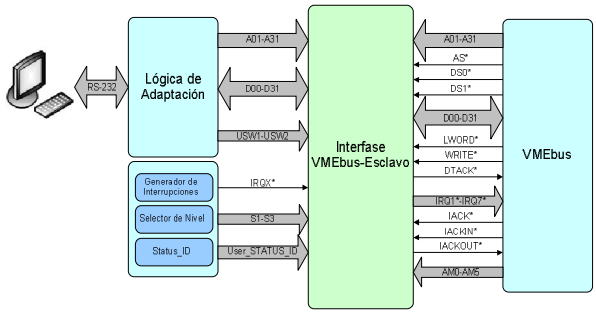

Figura 9: Esquema Implementado de la Interfaz.

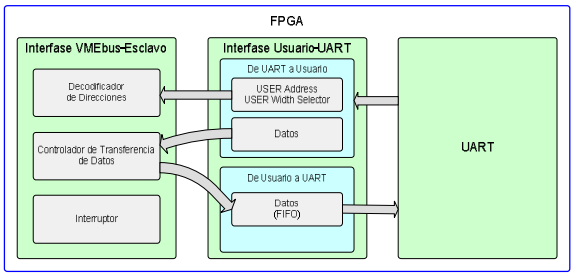

Figura 10: Lógica de adaptación serie.

En la Figura 11 se presenta el prototipo terminado en el la placa normalizada de VMEbus.

Figura 11: Prototipo Entrada - Salida VMEbus.

IV CONCLUSIONES

El objetivo de este trabajo fue cumplido al desarrollar una interfase capaz de manejar tanto las transferencias de datos como las interrupciones de un sistema VME, disponiendo solamente del estándar (Fig 12).

Se comprobó el funcionamiento de cada uno de los bloques diseñados para esta interfase, así como también el funcionamiento de la misma en forma integral con modulos externos.

Se ratificaron las características de la utilización de lenguaje de descripción de hardware generando una rápida solución (prototipado rápido), que ademas permite adaptarse a distintas prestaciones debido a su poder de reconfiguración, en particular cuando las características del problema a resolver está fuera del alcance tecnológico del momento. Muchas veces no se puede conseguir en el mercado los reemplazos o las adaptaciones necesarias para un sistema de instrumentación en utilización.

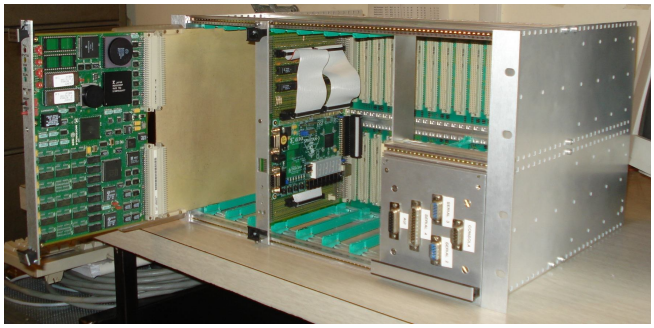

Figura 12: Prototipo en uso con placa procesadora SBC MVME 147 perteneciente al IAR

Agradecimiento: Al Instituto de Radioastronomía (IAR) que facilito el laboratorio para su implementación, especialmente a Martín Semegone y Martín Benitez quien intervinieron en el desarrollo.

REFERENCIAS

[1] VMEbus Especification manual Revision c.1. 1987.

[2] Foro VMEbus "*comp.arch.bus.vmebus*". http://groups.google.com/group/comp.arch.bus.vmebus

[3] Paterson, Wade D. "*The VMEbus Handbook*". VFEA International Trade Association (VITA). Ed.1993 http://www.vita.com/

[4] Davis, Leroy. "*VME Bus*". www.interfacebus.com/DesignConnectorVME.html.

[5] Skinner, Ed. "VMEbus for Software Engineers". http://www.flat5.net/vmebus.htm.

[6] 1076TM IEEE Standard VHDL Language Reference Manual. 2002.

[7] Yalamanchili. "Introductory VHDL: From Simulation To Syntesis". Prentice Hall. 2001

[8] Bergeron, Janick. "*Writing Testbenchs - Functional Verification of HDL Models*". Kluwer Academis Publishers. 2000

[9] Xilinx. "Spartan-3 FPGA Family: Complete Data Sheet".

[10] Bravo Muños, Ignacio. "Compilación y simulación de Cores de Xilinx en Modelsim v2.0". Universidad de Alcalá. Departamento de Electrónica. 2003

[11] Xilinx "Xilinx ISE/WebPack " Introduction to Schematic Capture and Simulation". 2003.. "Synthesis and Simulation Design Guide", "Constraints Guide 8.2.2", "CORE Generator Guide — ISE 5", "ISE 8.2 In-Depth Tutorial", "ISE 8.2i Quick Start Tutorial

[12] Mentor Graphics. "ModelSim Installation & Licensing, v6.1". 2005; "ModelSim Reference Manua, , Tutorial, User's Manua; v6.3c". 2007

[13] Achilli Federico Eduardo, Gil Santiago Andrés, "VMEbus", Cátedra de Trabajo Final, Departamento de Electrotecnia, UNLP, Facultad de Ingeniería.